Nama : Lorenzo Cartzens Seimima
Kelas : 2IA11
NPM : 50422812

Pengenalan PBO

Pemrograman berorientasi objek (Object Oriented Programming atau disingkat OOP) adalah paradigma pemrograman yang berorientasikan kepada objek yang merupakan suatu metode dalam pembuatan program, dengan tujuan untuk menyelesaikan kompleksnya berbagai masalah program yang terus meningkat. Objek adalah entitas yang memiliki atribut, karakter (bahavour) dan kadang kala disertai kondisi (state)

Perbedaan Pemrograman Berbasis Objek dan Pemrograman Terstruktur

- Pemrograman Berbasis Objek :
  - Menggabungkan fungsi dan data dalam kelas atau objek.
  - Memiliki ciri Encapsulation (pengemasan), Inheritance (penurunan sifat) dan
  - Polymorphism (perbedaan bentuk dan perilaku).
  - Struktur program ringkas, cukup dengan membuat Objek dan class.
  - Kode program sangat re-usable.

- Pemrograman Terstruktur :

• Memecah program dalam fungsi dan data.
• Memiliki ciri Sequence (berurutan), Selection (pemilihan) dan Repetition (perulangan).
• Struktur program rumit karena berupa urutan proses dan fungsi-fungsi.
• Re-use kode program kurang.

Objek dalam pemrograman

• Suatu Objek adalah unik, objek dapat mewakili sesuatu di dunia nyata, contohnya: orang, mobil, rekening bank dan lain sebagainya
* Objek mirip dengan suatu rekaman (record) dalam suatu sistem berkas (misalnya rekaman karyawan).
Suatu objek didefinisikan berdasarkan namanya atau menggunkan kata benda, objek juga memiliki atribut dan metode
• Objek adalah entitas yang memiliki atribut, karakter dan kadang kala disertai kondisi. Objek mempresentasikan sesuai kenyataan seperti siswa, mempresentasikan dalam bentuk konsep seperti merek dagang, juga bisa menyatakan visualilasi seperti bentuk huruf (font).

Dalam pengembangan perangkat lunak berorientasi objek, objek menyimpan state-nya dalam variabel dan perilaku dalam metode atau fungsi. Ini memungkinkan representasi yang lebih abstrak dan modular dalam membangun program.

Untuk membuat objek, bisa menggunakan kata kunci new yang digunakan untuk membuat objek baru, selanjutnya menentukan 3 langkah untuk membuat sebuah objek. Yaitu: mendeklarasikan variable, membuat objek baru (Instansiasi) dan pemanggilan konstruktor.

Contoh program :

```
public class Kendaraan{

    // Konstruktor Dengan Parameter
    public Kendaraan(String nama){
        System.out.println("Nama Kendaraannya Adalah "+ nama);
    }

    public static void main(String[]args){

        // Perintah untuk membuat objek jenis
        Kendaraan jenis = new Kendaraan("Pesawat Terbang");

    }

}

// Output = Nama Kendaraannya Adalah Pesawat Terbang
```

Pengertian Inheritance
Pewarisan adalah penggunaan kembali kelas yang ada untuk membuat kelas baru dengan menambahkan fitur baru. Setiap kelas turunan dapat menjadi induk bagi kelas turunan berikutnya. Constructor tidak diwariskan secara otomatis dalam pewarisan, kecuali menggunakan perintah "super". Java hanya mendukung pewarisan tunggal. Kelas yang memberikan warisan disebut superclass, sementara yang menerima disebut subclass. Untuk menerapkan inheritance dalam Java, gunakan "extends".
Contoh untuk memanggil superclass :

```
super()
super(parameter)
```

Contoh untuk memanggil atribut dan method milik superclass :

```
super.namaAtribut
super.namaMethod(parameter)
```

Pengertian Encapsulation
Enkapsulasi (encapsulation) merupakan cara untuk melindungi property (atribut) / method tertentu dari sebuah kelas agar tidak sembarangan diakses dan dimodifikasi oleh suatu bagian program.
Accessors :
Cara untuk melindungi data yaitu dengan menggunakan access modifiers (hak akses). Ada 4 hak akses yang tersedia, yaitu default, public, protected, private.

No	Modifier	Pada class dan interface	Pada method dan variabel
1	Default (tidak ada modifier)	Dapat diakses oleh yang sepaket	Diwarisi oleh subkelas dipaket yang sama, dapat diakses oleh method-method yang sepaket
2	Public	Dapat diakses dimanapun	Diwarisi oleh subkelasnya, dapat diakses dimanapun
3	Protected	Tidak bisa diterapkan	Diwarisi oleh subkelasnya, dapat diakses oleh method-method yang sepaket
4	private	Tidak bisa diterapkan	Tidak dapat dikases dimanapun kecuali oleh method-method yang ada dalam kelas itu sendiri

Aksesabilitas	public	private	protected	default
Dari kelas yang sama	Ya	Ya	Ya	Ya
Dari sembarang kelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari sembarang kelas di luar paket	Ya	Tidak	Tidak	Tidak
Dari subkelas dalam paket yang sama	Ya	Tidak	Ya	Ya
Dari subkelas di luar paket	Ya	Tidak	Ya	Tidak

Polymorphism

Overloading adalah diperbolehkannya dalam sebuah class memiliki lebih dari satu nama function/method yang sama tetapi memiliki parameter/argument yang berbeda

```
public class Overloading {
    public void Tampil(){
        System.out.println("I love Java");
    }
    public void Tampil(int i){
        System.out.println("Method dengan 1 parameter = "+i);
    }
    public void Tampil(int i, int j){
        System.out.println("Method dengan 2 parameter = "+i+" & "+j);
    }
    public void Tampil(String str){
        System.out.println(str);
    }

    public static void main(String a[]){
        Overloading objek = new Overloading();
        objek.Tampil();
        objek.Tampil(8);
        objek.Tampil(6,7);
        objek.Tampil("Hello world");
    }
}
```

Inheritance

Untuk setiap burung, ada satu set properti yang telah ditetapkan yang umum untuk semua burung dan ada satu set properti yang khusus untuk burung tertentu. Oleh karena itu, secara intuitif, kita dapat mengatakan bahwa semua burung mewarisi ciri-ciri umum seperti sayap, kaki, mata, dll. Oleh karena itu, dalam cara berorientasi objek untuk merepresentasikan burung, pertama-tama kita mendeklarasikan kelas “Bird” dengan seperangkat properti yang umum untuk semua burung. Dengan melakukan ini, kita dapat menghindari menyatakan sifat-sifat umum ini di setiap burung yang kita buat. Sebagai gantinya, kita cukup mewariskan kelas “Bird” di semua burung yang kita buat.

Encapsulation

Sekarang, kita telah mendefinisikan properti dari kelas “Bird” dan atribut yang dimiliki burung

seperti warna, sayap, kaki dapat diinisialisasi dengan membuat objek kelas burung. Namun, jika kita hanya dapat mengubah properti dari kelas “Bird” hanya dengan referensi dari objek, maka atribut kehilangan informasi yang awalnya diinisialisasi.
Misalnya, katakanlah kita awalnya membuat merpati dengan warna abu-abu dengan membuat konstruktor, setiap pengguna dengan instance objek merpati dapat mengubah warna ini menjadi merah atau hitam hanya dengan merujuk atribut dengan kata kunci “this". Untuk menghindari hal ini, kita menyertakan properti ke dalam sebuah metode. Metode ini disebut getter dan setter atribut. Idenya adalah untuk hanya menyertakan inisialisasi dan pengambilan atribut dalam suatu metode alih-alih langsung merujuk atribut secara langsung. Ini juga memberikan keuntungan karena setter memberi kita kendali penuh dalam menetapkan nilai ke atribut dan membantu kita membatasi perubahan yang tidak perlu.

```
// Implementing the bird class
public class Bird {

    // Few properties which
    // define the bird
    String animal = "All Birds";
    String color;
    int legs;

    // Implementing the getters and
    // setters for the color and legs.

    public void setColor(String color)
    {
        this.color = color;
    }

    public String getColor()
    {
        return this.color;
    }

    public void setLegs(int legs)
    {
        this.legs = legs;
    }

    public int getLegs()
    {
        return this.legs;
    }

    // Few operations which the
    // bird performs
    public void eat()
    {
        System.out.println(
            "This bird has eaten");
    }

    public void fly()
    {
        System.out.println(
            "This bird is flying");
    }
}
```

Polymorphism
Sekarang, kita ingin membuat merpati dapat terbang dengan menggunakan method “fly” namun kita ingin mendefinisikan tujuan terbang merpati tersebut dan kita ingin mendefinisikan makanan yang dimakan oleh merpati tersebut. Kita dapat melakukannya dengan menggunakan konsep overload method.

```java
// Creating the Pigeon class which
// extends the bird class
public class Pigeon extends Bird {

    String animal ="Pigeon";
    // Overriding the fly method
    // which makes this pigeon fly
    @Override
    public void fly() //override method
    {
        System.out.println(
            "Pigeon flys!!!");
    }

    //overload method to method in this class
    public void fly(String place){
        System.out.println("Pegeon flys to "+ place);
    }

    //overload method to method in parent class
    public void eat(String food)
    {
        System.out.println(
            "Pigeon eats "+food);
    }

    //super & this
    public void about(){
        System.out.println(super.animal+" live freely in the sky");
        System.out.println(this.animal+ " can recognise each letter of the human
    }
}
```

Testing

Setelah kita selesai membuat kelas “Bird” dan kelas “Pigeon”, kita perlu membuat sebuat kelas untuk menjalankan atau memanggil kelas-kelas tersebut beserta method-methodnya.

```java
//encapsulation, inheritance, & polymorphism testing
public class OOPTest {
    public static void main(String args[]){
        Bird bird = new Bird();
        Pigeon pigeon = new Pigeon();

        bird.eat();
        bird.fly();

        System.out.println("");

        //polymorphism
        pigeon.eat("Seed");
        pigeon.fly();
        pigeon.fly("Sky");

        System.out.println("");

        //Mutators & inheritance
        pigeon.setColor("grey");
        pigeon.setLegs(2);
        System.out.println(
                "this pigeon has "+pigeon.getColor()+" color "
                + "and "+ pigeon.getLegs()+" legs");
        pigeon.eat();

        System.out.println("");
        pigeon.about();

    }
}
```